ECON457 lab01 R Basic

Jon

September 13, 2016

Some basic information related to R lab01.

## Data type

R has five basic or "atomic" classes of objects:

character

numeric (real numbers)

integer

complex

logical (True/False)

x <- c(0.5, 0.6) ## numeric  
x <- c(TRUE, FALSE) ## logical  
x <- c(T, F) ## logical  
x <- c("a", "b", "c") ## character  
x <- c(1+0i, 2+4i) ## complex  
x <- 9:29 ## integer

## Data Structure

### Vector

The most basic type of R object is a vector. Empty vectors can be created with the vector() function. There is really only one rule about vectors in R, which is that A vector can only contain objects of the same class.

You can also use the vector() function to initialize vectors.

x <- vector("numeric", length = 10)   
x

## [1] 0 0 0 0 0 0 0 0 0 0

**A vector can only contain objects of the same class.**

### Matrix

m <- matrix(nrow = 2, ncol = 3)   
m

## [,1] [,2] [,3]  
## [1,] NA NA NA  
## [2,] NA NA NA

Matrices can be created by column-binding or row-binding with the cbind() and rbind() functions.

x <- 1:3  
y <- 10:12  
cbind(x, y)

## x y  
## [1,] 1 10  
## [2,] 2 11  
## [3,] 3 12

rbind(x, y)

## [,1] [,2] [,3]  
## x 1 2 3  
## y 10 11 12

### Data Frames

Data frames are used to store tabular data in R. They are an important type of object in R and are used in a variety of statistical modeling applications. Hadley Wickham's package dplyr has an optimized set of functions designed to work efficiently with data frames.

x <- data.frame(foo = 1:4, bar = c(T, T, F, F))   
x

## foo bar  
## 1 1 TRUE  
## 2 2 TRUE  
## 3 3 FALSE  
## 4 4 FALSE

nrow(x)

## [1] 4

ncol(x)

## [1] 2

names(x)

## [1] "foo" "bar"

|  |  |  |
| --- | --- | --- |
| Object | Set column names | Set row names |
| data frame | names() | row.names() |
| matrix | colnames() | rownames() |

### Indexing

Indexing is used to specify the elements of an array. Indexing also allows you to get out certain bits of information from an array. To index into an array, type the name of the array, followed by the index of the element you want in brackets. Note that in R, indices start at 1.

For a multidimensional array, index by [row,column]

To index an entire row or column, use a colon.

Below we index into the named y to get out the element in the second row, third column, 6.

y <- c( 1,2,3, 4,5,6)  
y <- matrix(y, nrow = 2, ncol = 3, byrow = T)  
y[2,3]

## [1] 6

Below, we index the entire first row of the array named y.

y[1,]

## [1] 1 2 3

Below, we index the entire second column of the array named y.

y[,2]

## [1] 2 5

## Load Data from Csv File

R works with many data formats.

csv file is the most convenient.

Load data locally

Download file from [here](https://github.com/Eighty20/eighty20.github.io/raw/master/_rmd/Post_data/All_pokemon.csv)

#pokemon <- read.csv("All\_pokemon.csv")

# require(RCurl)  
# tempfile()  
# remember delete the original pokemon.csv or comment this line  
# temporaryFile <- tempfile() # "pokemon.csv"  
# download.file("https://github.com/Eighty20/eighty20.github.io/raw/master/\_rmd/Post\_data/All\_pokemon.csv",destfile=temporaryFile, method = "internal")  
# pokemon <- read.csv(temporaryFile)

or directly from online repository.

pokemon <- read.csv("https://github.com/Eighty20/eighty20.github.io/raw/master/\_rmd/Post\_data/All\_pokemon.csv")

head(pokemon)

## Nat Pokemon HP Atk Def SA SD Spd Total Type.I Type.II Gender  
## 1 1 Bulbasaur 45 49 49 65 65 45 318 Grass Poison M (87.5%)  
## 2 2 Ivysaur 60 62 63 80 80 60 405 Grass Poison M (87.5%)  
## 3 3 Venusaur 80 82 83 100 100 80 525 Grass Poison M (87.5%)  
## 4 4 Charmander 39 52 43 60 50 65 309 Fire M (87.5%)  
## 5 5 Charmeleon 58 64 58 80 65 80 405 Fire M (87.5%)  
## 6 6 Charizard 78 84 78 109 85 100 534 Fire Flying M (87.5%)  
## Evolves.From lvl\_up Evolves.Into  
## 1 -- Lv. 16 Ivysaur  
## 2 Bulbasaur Lv. 32 Venusaur  
## 3 Ivysaur -- --  
## 4 -- Lv. 16 Charmeleon  
## 5 Charmander Lv. 36 Charizard  
## 6 Charmeleon -- --

str(pokemon)

## 'data.frame': 251 obs. of 15 variables:  
## $ Nat : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ Pokemon : Factor w/ 250 levels "Abra","Aerodactyl",..: 17 93 235 23 24 22 208 240 15 19 ...  
## $ HP : int 45 60 80 39 58 78 44 59 79 45 ...  
## $ Atk : int 49 62 82 52 64 84 48 63 83 30 ...  
## $ Def : int 49 63 83 43 58 78 65 80 100 35 ...  
## $ SA : int 65 80 100 60 80 109 60 65 85 20 ...  
## $ SD : int 65 80 100 50 65 85 54 80 105 20 ...  
## $ Spd : int 45 60 80 65 80 100 43 58 78 45 ...  
## $ Total : int 318 405 525 309 405 534 314 405 530 195 ...  
## $ Type.I : Factor w/ 16 levels "Bug","Dark","Dragon",..: 8 8 8 6 6 6 16 16 16 1 ...  
## $ Type.II : Factor w/ 15 levels "","Dark","Dragon",..: 11 11 11 1 1 7 1 1 1 1 ...  
## $ Gender : Factor w/ 7 levels "50/50","F (100%)",..: 6 6 6 6 6 6 6 6 6 1 ...  
## $ Evolves.From: Factor w/ 120 levels "--","Alakazam",..: 1 7 43 1 10 11 1 103 115 1 ...  
## $ lvl\_up : Factor w/ 57 levels "","--","Dusk Stone",..: 17 30 2 17 34 2 17 34 2 40 ...  
## $ Evolves.Into: Factor w/ 130 levels "--","Alakazam",..: 51 121 1 15 14 1 124 11 1 72 ...

More information about [pokeman](http://www.levelbased.com/guides/pokemon-go/pokemon/?q=)



Bulbasaur

## Methods on Objects

A method is a procedure associated with an object. Below is a list of common methods used on arrays.

length(y)

## [1] 6

dim(y)

## [1] 2 3

str(y)

## num [1:2, 1:3] 1 4 2 5 3 6

attributes(y)

## $dim  
## [1] 2 3

typeof(y)

## [1] "double"

head(y, 1)

## [,1] [,2] [,3]  
## [1,] 1 2 3

tail(y,1)

## [,1] [,2] [,3]  
## [2,] 4 5 6

## Mathematical Operations

Below is a list of common mathematical operations that you can perform on numerical types.

x + y performs addition

x - y performs subtraction

x \* y performs multiplication

x / y performs division

x ^ y raises x to the yth power

x = y assigns the variable named "x" to the value y

x == y evaluates to a Boolean, true if x equals y, false otherwise

x != y evaluates to a Boolean, true if x does not equal y, false otherwise

x > y evaluates to a Boolean, true if x is greater than y, false otherwise

x < y evaluates to a Boolean, true if x is less than y, false otherwise

x <= y evaluates to a Boolean, true if x is less than or equal to y, false otherwise

x >= y evaluates to a Boolean, true if x is greater than or equal to y, false otherwise

9-4

## [1] 5

9 / 4

## [1] 2.25

log(exp(10))

## [1] 10

sin(pi/2)

## [1] 1

## Control Flow

### Condition if/elseif/else

Conditional evaluation allows portions of code to be evaluated or not evaluated depending on the value of a Boolean expression. You do not need all if/elseif/else statements. You can have conditional evaluations with just an if, or just an if/else.

The general structure of conditional evaluation is as follows.

After assigning values to x and y and running the code we obtain the following output.

x =1  
y=2  
if (x<y){  
 print(x)  
}else{  
 print(y)  
}

## [1] 1

### For Loops

A for loop allows you to specify the number of iterations for the repeated execution of a code block. They are great when you know how many iterations you want to run.

The general form of a for loop is shown below. The example shows a for loop that calculates the sum of the integers 1 through 10 and prints the final result.

Note that to obtain a range of integers, we use the colon : symbol.

sum = 0  
for(num in 1:10){  
 sum = sum + num  
}  
print(sum)

## [1] 55

A range of integers, we use the colon : symbol.

1:10

## [1] 1 2 3 4 5 6 7 8 9 10

### A series of video of R tutorial.

Each tutorial usually only is 2 minutes long.

<http://www.twotorials.com/>

See what packages we used.

sessionInfo()

## R version 3.2.1 (2015-06-18)  
## Platform: x86\_64-w64-mingw32/x64 (64-bit)  
## Running under: Windows 7 x64 (build 7601) Service Pack 1  
##   
## locale:  
## [1] LC\_COLLATE=English\_Canada.1252 LC\_CTYPE=English\_Canada.1252   
## [3] LC\_MONETARY=English\_Canada.1252 LC\_NUMERIC=C   
## [5] LC\_TIME=English\_Canada.1252   
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## loaded via a namespace (and not attached):  
## [1] magrittr\_1.5 tools\_3.2.1 htmltools\_0.3.5 yaml\_2.1.13   
## [5] Rcpp\_0.12.7 stringi\_1.1.1 rmarkdown\_1.0 knitr\_1.14   
## [9] stringr\_1.1.0 digest\_0.6.10 evaluate\_0.9